View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Calhoun, Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Faculty and Researcher Publications

Faculty and Researcher Publications

2007-03

Anti-Forensics: Techniques, Detection and Countermeasures

Garfinkel, Simson

Garfinkel, S., "Anti-Forensics: Techniques, Detection and Countermeasures", The 2nd International Conference on i-Warfare and Security (ICIW), Naval Postgraduate School,

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

Anti-Forensics: Techniques, Detection and Countermeasures

Simson Garfinkel
Naval Postgraduate School, Monterey, CA, USA
simsong@acm.org

Abstract: Computer Forensic Tools (CFTs) allow investigators to recover deleted files, reconstruct an intruder's activities, and gain intelligence about a computer's user. Anti-Forensics (AF) tools and techniques frustrate CFTs by erasing or altering information; creating "chaff" that wastes time and hides information; implicating innocent parties by planting fake evidence; exploiting implementation bugs in known tools; and by leaving "tracer" data that causes CFTs to inadvertently reveal their use to the attacker. Traditional AF tools like disk sanitizers were created to protect the privacy of the user. Anti-debugging techniques were designed to protect the intellectual property of compiled code. Rootkits allow attackers to hide their tools from other programs running on the same computer. But in recent years there has been an emergence of AF that directly target CFTs. This paper categorizes traditional AF techniques such as encrypted file systems and disk sanitization utilities, and presents a survey of recent AF tools including Timestomp and Transmogrify. It discusses approaches for attacking forensic tools by exploiting bugs in those tools, as demonstrated by the "42.zip" compression bomb. Finally, it evaluates the effectiveness of these tools for defeating CFTs, presents strategies for their detection, and discusses countermeasures.

Keywords: 42.zip; anti-computer forensics; cybercrime; EnCase; hacker tools; privacy enhancing tools.

1. Background: Computer forensics and anti-forensics

Computer forensics is a new discipline that uses "scientific knowledge for collecting, analyzing, and presenting evidence to the courts" (USCERT 2005). Computer forensic tools (CFTs) assist forensic examiners by collecting information from a computer system; making a true and permanent copy of that information, so that it can be used in a legal proceeding; and analyzing data to uncover information that may not be immediately obvious.

CFTs fall broadly into two classes. *Persistent data tools* (e.g. The Sleuth Kit (Carrier 2006)) analyze data that is stored and that remains when a computer system is turned off. *Volatile data tools* (e.g. NetIntercept (Sandstorm 2006)) analyze information that is transitory and would be lost if not captured, such as the contents of a computer's memory or the flow of packets as they transverse a network.

"Anti-forensics" (AF) is a growing collection of tools and techniques that frustrate forensic tools, investigations and investigators.

Liu and Brown identify four primary goals for anti-forensics:

- Avoiding detection that some kind of event has taken place.
- Disrupting the collection of information.
- Increasing the time that an examiner needs to spend on a case.
- Casting doubt on a forensic report or testimony (Liu and Brown, 2006).

Other goals might include:

- Forcing the forensic tool to reveal its presence.
- Subverting the forensic tool (e.g., using the forensic tool itself to attack the organization in which it is running).
- Mounting a direct attack against the forensic examiner (e.g., discovering and disconnecting the examiner's network, or bombing the building in which the examiner is working).
- Leaving no evidence that an anti-forensic tool has been run.

There is growing interest in AF tools and techniques (AFTs). AF researchers have developed and distributed tools freely on the Internet. Some developers say their work will force improvements in existing forensic tools and process (Liu and Brown, 2006). AFTs also demonstrate the limitations of today's computer forensic techniques, challenging the so-called "presumption of reliability" that courts have afforded tools such as EnCase (Buskirk and Liu, 2006). Some claim that "the lack of true innovation in the forensics world is because there's no pressure to do so" and that they are "not creating vulnerabilities, just identifying them"

(Foster and Liu, 2005). But an unfortunate side-effect is that these tools may be used to "exonerate a guilty party" or "implicate an innocent party by planting data" (Liu and Brown, 2006).

1.1 Outline of this paper

Section 2 of this paper discusses traditional AF techniques such as overwriting, cryptography and steganography. Section 3 discusses AF techniques that work by minimizing the attacker's footprint; Section 4 discusses techniques that directly attack CFTs. Section 5 discusses techniques designed to unmask the use of CFTs. Section 6 concludes.

2. Traditional anti-forensics

Tools that overwrite information that might be the subject of an investigation are the oldest and most common forms of anti-forensic tools available today. Such tools are easy to write and validate, require little training to run, and are distributed with most operating systems.

2.1 Overwriting data and metadata

There exist many programs for overwrite useful information on a storage device so that it is difficult or impossible to recover. These programs can overwrite data, metadata, or both.

2.1.1 Modes of operation

Overwriting programs typically operate in one of three modes:

- The program can overwrite the entire media.
- The program can attempt to overwrite individual files. This task is complicated by journaling file systems: the file itself may be overwritten, but portions may be left in the journal.
- The program can attempt to overwrite files that were previously "deleted" but left on the drive. Programs typically do this by creating one or more files on the media and then writing to these files until no free space remains, taking special measures to erase small files — for example, files that exist entirely within the Windows Master File Table of an NTFS partition (Garfinkel and Malan, 2005).

Programs employ a variety of techniques to overwrite data. Apple's Disk Utility allows data to be overwritten with a single pass of NULL bytes, with 7 passes of random data, or with 35 passes of data. Microsoft's **cipher.exe**, writes a pass of zeros, a pass of FFs, and a pass of random data, in compliance with DoD standard 5220.22-M. (US DoD, 1995). In 1996 Gutmann asserted that it might be possible to recover overwritten data and proposed a 35-pass approach for assured sanitization (Gutmann 1996). However, a single overwriting pass is now viewed as sufficient for sanitizing data from ATA drives with capacities over 15 GB that were manufactured after 2001 (NIST 2006).

The real challenge that the authors of sanitizing tools face not the act of overwriting, but determining which data to overwrite. Although it is relatively easy to overwrite an entire hard drive (#1 above), it can be extremely difficult to selectively overwrite sensitive information while leaving the rest. Geiger evaluated commercial anti-forensic tools and found that many left sensitive information (Geiger 2005). Many tools also left telltale evidence indicating which tool had been run, potentially exposing the user's actions.[1]

2.1.2 Overwriting metadata

If the examiner knows when an attacker had access to a Windows, Mac or Unix system, it is frequently possible to determine which files the attacker accessed, by examining file "access" times for every file on the system. Some CFTs can prepare"timeline" of the attacker's actions by sorting all of the computer's timestamps in chronological order. Although an attacker could wipe the contents of the media, this action itself might attract attention. Instead, the attacker might hide her tracks by overwriting the access times themselves so that the timeline could not be reliably constructed.

For example, Timestomp will overwrite NTFS "create," "modify," "access," and "change" timestamps (Metasploit 2006). The Defiler's Toolkit can overwrite inode timetamps and deleted directory entries on many

[1] In March 2006, the US Court of Appeals for the 7th Circuit ruled that Jacob Citrin's use of a secure file erase program violated the Computer Fraud and Abuse Act. The Court held that the use of the secure deletion program constituted the willful destruction of information on a computer that Critin was no longer authorized to use (Andersen 2006).

Unix systems; timestamps on allocated files can also be modified using the Unix **touch** command (Grugq 2003).

An alternative to overwriting metadata is for the attacker to access the computer in such a way that metadata is not created. For example, a partition can be mounted read-only or accessed through the raw device to prevent the file access times from being updated. The Windows registry key HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate can be set to "1" to disable updating of the last-accessed timestamp; this setting is default under Windows Vista (Microsoft 2006).

2.2 Cryptography, steganography, and other data hiding approaches

Cryptography, steganography and other data hiding approaches can be used effectively against most forensic approaches. But while cryptography is very effective at hiding information, encrypted data itself is easy to detect: encrypted data has exceptionally high entropy, and many products embed specific flags, headers or other signatures in their encrypted data. Some forensic tools can decrypt encrypted data if the key is obtained—for example, through spyware or other covert channels. Even if the original plaintext data cannot be recovered, the mere fact that encryption has been used may attract unwanted attention to the attacker.

2.2.1 Cryptographic file systems

Cryptographic file systems transparently encrypt data when it is written to the disk and decrypt data when it is read back, making the data opaque to any attacker (or CFT) that does not have the key. These file systems are now readily available for Windows, Mac OS, and Linux. The key can be protected with a passphrase or stored on an auxiliary device such as a USB token. If there is no copy of the key, intentionally destroying the key makes the data stored on the media inaccessible (Boneh and Lipton, 1996). Even if the cryptographic system lacks an intentional sanitization command or "self-destruct," cryptography can still be a potent barrier to forensic analysis if the cryptographic key is unknown to the examiner.

Cryptography can also be used at the application level. For example, Microsoft Word can be configured to encrypt the contents of a document by specifying that the document has a "password to open." Although older versions of Microsoft Word encrypted documents with a 40-bit key that can be cracked with commercial tools, modern versions can optionally use a 128-bit encryption that is uncrackable if a secure passphrase is used.

2.2.2 Encrypted network protocols

Network traffic can likewise be encrypted to protect its content from forensic analysis. Cryptographic encapsulation protocols such as SSL and SSH only protect the content of the traffic. Protecting against traffic analysis requires the use of intermediaries. Onion Routing (Goldschlag, Reed and Syverson, 1999) combines both approaches with multiple layers of encryption, so that no intermediary knows both ends of the communication and the plaintext content.

2.2.3 Program packers

Packers are commonly used by attackers so that attack tools will not be subject to reverse engineering or detection by scanning. Packers such as PECompact (Bitsum 2006) and Burneye (Vrba 2004) will take a second program, compress and/or encrypt it, and wrap it with a suitable extractor. Packers can also incorporate active protection against debugging or reverse engineering techniques. For example, Shiva will exit if its process is being traced; if the process is not being traced, it will create a second process, and the two processes will then trace each other, since each process on a Unix system may only be traced by one other process. (Mehta and Clowes, 2003)

Packed programs that require a password in order to be run can be as strong as their encryption and password. However, the programs are vulnerable at runtime. Burndump is a loadable kernel module (LKM) that automatically detects when a Burneye-protected file is run, waits for the program to be decrypted, and then writes the raw, unprotected binary to another location (ByteRage 2002). Packed programs are also vulnerable to static analysis if no password is required (Eagle 2003).

2.2.4 Steganography

Steganography can be used to embed encrypted data in a *cover text* to avoid detection. Steghide embeds text in JPEG, MBP, MP3, WAV and AU files (Hetzl 2002). Hydan exploits redundancy in the x86 instruction set; it can encode roughly 1 byte per 110 (El-Khalil 2004). Stegdetect (Provos 2004) can detect some forms of steganography. StegFS hides encrypted data in the unused blocks of a Linux ext2 file system, making the data "look like a partition in which unused blocks have recently been overwritten with random bytes using some disk wiping tool" (McDonald and Kuhn, 2003). TrueCrypt allows a second encrypted file system to be hidden within the first (TrueCrypt 2006). The goal of this filesystem-within-a-filesystem is to allow the TrueCrypt users to have a "decoy" file system with data that is interesting but not overtly sensitive. A person who is arrested or captured with a TrueCrypt-protected laptop could then give up the first file system's password, with the hope that the decoy would be sufficient to satisfy the person's interrogators.

2.2.5 Generic data hiding

Data can also be hidden in unallocated or otherwise unreachable locations that are ignored by the current generation of forensic tools.

Metasploit's Slacker will hide data within the slack space of FAT or NTFS file system. FragFS hides data within the NTFS Master File Table. RuneFS (Grugq 2003) stores data in bad blocks. (Thompson and Monroe, 2006). Waffen FS stores data in the ext3 journal file (Eckstein and Jahnke 2005). KY FS stores data in directories (Grugq 2003). Data Mule FS stores data in inode reserved space (Grugq 2003). It is also possible to store information in the unallocated pages of Microsoft Office files.

Information can be stored in the Host Protected Area (HPA) and the Device Configuration Overlay (DCO) areas of modern ATA hard drives. Data in the HPA and DCO is not visible to the BIOS or operating system, although it can be extracted with special tools. Data can be hidden from network monitoring by encoding it in TCP sequence numbers, window sizes, unused TCP or IP options, or even introducing intentional jitter in packet timing (Cabuk, Brodley and Shields, 2004). With this taxonomy, rootkits can be considered as simply another data hiding technique that hides information about files or active processes from other programs running on the same computer (Brumley 2000). As with the HPA, all of these techniques for data hiding can be detected with tools that understand the typical format of file system or application structures and report when unused or allocated regions are anomalous. For example, The Defiler's Toolkit has provisions to hide data in "bad blocks" of ext2fs file systems. This is effective against The Coroner's Toolkit, which does not look inside blocks that are labeled "bad," but does not work against The Sleuth Kit (Carrier 2006), an updated version of TCT (Wonko-Ga 2004).

3. AF techniques that minimize footprint

Many of the techniques in Section 2 are easy to detect when they are used, provided that the examiner knows where to look. Another approach is to minimize the "footprint," or data that the attacker has left behind. In this way, there is less data for the CFT to analyze.

3.1 Memory injection and Syscall Proxying

Buffer overflow exploits allow an attacker to inject and run code in the address space of a running program, effectively changing the victim program's behavior. Traditionally buffer overflows have just been used as a point-of-entry to a remote system, after which the attacker uploads tools that are then saved on the target machine's hard drive.

The "Userland Execve" technique (Grugq 2003a, Pluf & Ripe 2006) allows programs on the victim computer to be loaded and run without the use of the Unix execve() kernel call, allowing the attacker to overcome kernel-based security systems that may deny access to execve() or log its use to a secure logging service. The ul_exec'ed program can even be read from a network without ever being written to the victim's disk. Instead of uploading an entire exploit program, the attacker can upload a *system call proxy* that accepts remote procedure calls from the attacker's computer, executes the requested system call on the victim's machine, and sends the result back to the attacker. This approach has the advantage that the attacker's tools never need to be uploaded at all to the compromised machine, but its disadvantage of increasing (perhaps dramatically) the amount of network traffic between the compromised machine and the attacker — as well as introducing potentially problematic latency. System call proxying is implemented by Impact, a commercial penetration tool sold by Core Security Technologies.

3.2 Live cds, bootable USB tokens and virtual machines

Live CDs, bootable USB tokens, and virtual machines give attackers a tool for running programs of their choice while simultaneously containing the spread of forensically-useful information on the attacker's system.

- **Live CDs** are an operating system distribution that boots and runs from a read-only device. Live CDs typically have a window system, web browser and SSH client, and run with virtual memory disabled.
- **Bootable USB tokens** are similar to a Live CD except that the operating system is contained within an attachable USB device. These tokens can typically store more information that CDs and allow information to be saved, generally via encryption.
- **Virtual M**are "client" operating systems run inside a virtualization program such as VMWare Player, Parallels, or Microsoft Virtual PC. These systems typically store all of the states associated with the client operating system within a small set of files on the host computer.

Using a Live CD or Bootable USB Token, for example, an attacker could boot a copy of Linux on a PC provided by a library, use that PC to attack a series of computers, then turn off the computer and walk away: no trace of the attack would be left on the computer for later investigative analysis. Virtualization software makes it possible to perpetrate the attack without even rebooting the host computer. Following the attack, the attacker would only need to securely erase the files associated with the virtual machine. Virtual machines can also be used directly by the attacker on the victim's machine as a kind of super-rootkit. CFTs running in the VM would not see the super-rootkit, because it would be running outside the VM (Rutkowska 2004) (King et al 2006).

3.3 Anonymous identities and storage

Attackers have long protected their actual identity by making use of anonymous accounts available at Hotmail, Yahoo and Gmail. Recently the amount of storage associated with these accounts has been dramatically increased. Attackers can utilize this storage to avoid the risk of storing attack tools and captured information on their own computers (Dear 2005).

4. AF techniques that exploit CFT bugs

If an attacker has access to a CFT or knowledge of how that tool works, the attacker can craft data that will manifest bugs within the CFT. Properly triggered, these bugs can accomplish many anti-forensic goals.

4.1 Failure to validate data

As with other programs, CFTs that do not properly validate their input data can potentially be subverted—for example, by a buffer overflow attack. For example, CVE CAN-2001-1279 details a vulnerability in tcpdump's decoding routines for AFS RPC packets which then can be subverted by an attacker and used to run arbitrary code (CERT 2002). Similar problems have been discovered in Snort (Infoworld 2003) and Ethereal (iDefense 2005). This kind of vulnerability in a network forensics analysis tool (NFAT) is particularly easy to exploit, because the NFAT is exposed to a potentially unlimited amount of information from an attacker.

4.2 Denial of service attacks

Any CFT resource (memory, CPU, etc.) whose use is determined by input data is potentially subject to a denial-of-service attack. For example, certain Windows logfile analysis tools will attempt to execute regular expressions that are embedded in log file entries; carefully constructed regular expressions can be used to cause these tools to hang when they are executed (Foster and Liu, 2005). Compression bombs are a special category of DoS attacks on CFTs and other tools that attempt to analyze the content of container files. These bombs are small data files that consumes a tremendous amount of storage when uncompressed. For example, 42.zip is a 43,374 byte file that contains 16 zipped files, each of which contains 16 zipped files, and so on, for a total of 4TB of data (SANS 2001).

4.3 Fragile heuristics

CFTs frequently need to determine the type of file or data object to allow for efficient processing. For example, a forensic examiner may try to save time by omitting the contents of executable files from searches. Many tools determine file type by simply consulting the file's extension and the first few bytes of the file's contents (the "magic number"). Attackers who know the heuristics that a CFT uses for identifying files can exploit them. For example, EnCase will classify a Windows file as executable if it has an extension

of ".exe" and contains the letters "MZ" as the first two characters of the file (Liu and Stach, 2006). The Metasploit Project's Transmogrify program can turn a text file into an "executable" by changing a text file's extension from .txt to .exe and putting the letters "MZ" at the beginning of the file, EnCase will think that the file is a binary and not scan it.

5. AFTs that detect CFTs

AFTs can change their behavior if they can detect that a CFT is in use. For example, a packer might not decrypt its payload if it realizes that it is running on a disk that has been imaged. A worm might refuse to propagate if it discovers that a network is being surveilled.

5.1 Countering forensic analysis with SMART

Self-Monitoring, Analysis and Reporting Technology (SMART) built into most hard drives today report the total number of power cycles (Power_Cycle_Count), the total time that a hard drive has been in use (Power_On_Hours or Power_On_Minutes), a log of high temperatures that the drive has reached, and other manufacturer-determined attributes. These counters can be reliably read by user programs and cannot be reset. Although the SMART specification implements a DISABLE command (SMART 96), experimentation indicates that the few drives that actually implement the DISABLE command continue to keep track of the time-in-use and power cycle count and make this information available after the next power cycle (McLeod 2005). AFTs can read SMART counters to detect attempts at forensic analysis and alter their behavior accordingly. For example, a dramatic increase in Power_On_Minutes might indicate that the computer's hard drive has been imaged (McLeod 2005).

5.2 Detecting network forensics

Network forensics requires some kind of network monitoring. There are two primary techniques for detecting this monitoring: detecting hosts that have their Ethernet interfaces in "promiscuous" mode, and observing when information captured from a monitored network is used.

5.2.1 Detecting hosts in "Promiscuous" mode

Many network forensics systems capture information using an Ethernet interface that in promiscuous mode—that is, the interface captures all packets on the local area network, rather than just those addressed to the host. Properly configured network monitors should be unable to transmit on the network that is being monitored. Monitoring systems are frequently not configured this way. These systems can be detected by the way that they respond to various kinds of malformed IP packets (Sanai 2001).

5.2.2 Detection of monitoring with DNS

Another way to detect network monitoring is for an attacker to send packets across a network that have as their destination an Ethernet and IP address that is on the subnet but not currently in use, and have a source address that comes from a rarely used network. Upon seeing such packets, many network monitoring tools will initiate a reverse DNS request attempting to resolve the hostname of the rarely used network. If the attacker can monitor the DNS server that would handle such requests, the attacker may infer that monitoring of packets or network flows is taking place. This technique can work if there is a flaw in the CFT, or simply if the CFT is used improperly (Rudys 2000).

6. Countermeasures and conclusion

Many of the anti-forensic techniques discussed in this paper can be overcome through improved monitoring systems or by fixing bugs in the current generation of computer forensic tools. Overwriting tools can be frustrated by positioning data so that the attacker does not have the ability to overwrite it—for example, by sending log files to a "log host" or CD-R, assuming that the attacker does not have physical access. Weak file identification heuristics can be replaced with stronger ones. Compression bombs can be defeated with more intelligent decompression libraries. Although there is anecdotal evidence that file encryption and encrypted file systems are beginning to pose a problem for law enforcement, there are also many reports of officers being able to recover cryptographic passwords and keys using spyware, keyboard loggers, and other tactics. The prudent attacker is safer using a sanitization tool than a cryptographic one, because the sanitizer actually destroys information.

Many of the techniques discussed in this paper still appear limited to the research community, although there are occasional reports of specific tools being used by technically sophisticated computer criminals. Because law enforcement resources are limited, it seems reasonable to hypothesize that, other things being equal; attackers employing anti-forensic technology are less likely to be apprehended than those who do not. Because the goal of anti-forensic technology is specifically to confound investigations, it is possible that their use and perhaps even possession will be banned in some organizations. However, the inclusion of high-quality anti-forensic technology in consumer operating systems to promote data privacy goals are sure to make such bans futile. Computer forensics has traditionally relied upon information that was inadvertently left behind by other programs. Organizations soon may need to decide explicitly what information they wish to preserve as part of ordinary operations, and then make arrangements to preserve that information in a forensically sound manner.

Acknowledgements

The author wishes to thank Steven Bauer, Chris Eagle, Kevin Fu, Matthew Geiger, Jonathan Herzog, Beth Rosenberg and Vincent Liu for their valuable comments on a previous draft of this paper.

References

Andersen (2006), "International Air Centers, LLC, v. Jacob Citrin", 05-1522, Argued October 24, 2006, decided March 8, 2006. US Court of Appeals for the 7th Circuit.

Bitsum (2006), "PECompact: For maximum compression and speed," [online] http://www.bitsum.com/pec2.asp

Boneh and Lipton (1996), "A Revocable Backup System," Proceedings of the Sixth USENIX UNIX Security Symposium. [online] http://www.usenix.org/publications/library/proceedings/sec96/boneh.html

Brumley (2002), "Rootkits: How Intruders Hide", 23 Feb. [online] http://www.globalsecurity.org/intell/library/news/2000/02/000223-hack1.htm

Buskrik and Liu (2006), "Digital Evidence: Challenging the Presumption of Reliability", *Journal of Digital Forensic Practice*, 1:19—26, 2006. DOI 10.1080/15567280500541421

ByteRage (2002), "TESO Burneye Unwrapper", July 13. [online] http://www.securiteam.com/tools/5BP0H0U7PQ.html

Cabuk, Brodley and Shields (2004), "IP covert timing channels: design and detection," Proceedings of the 11th ACM conference on Computer and Communications Security, pp. 178-187

Carrier (2006) The Sleuth Kit [online] http://www.sleuthkit.org/

CERT (2002), "Vulnerability Note VU#797201: tcpdump vulnerable to buffer overflow via improper decoding of AFS RPC (Rx) packets," June 7. [online] https://www.kb.cert.org/vuls/id/797201.

Dear (2005), "An Exploration of Future Anti-Forensic Techniques" [online] http://www.assuremind.com/antiForensics.pdf

Eagle, C. (2003), "Strike/Counter-Strike: Reverse Engineering Shiva," Black Hat Federal. http://www.blackhat.com/presentations/bh-federal-03/bh-federal-03-eagle/bh-fed-03-eagle.pdf

Eckstein and Jahnke (2005), "Data Hiding in Journaling File Systems", *DFRWS 2005*. http://www.dfrws.org/2005/proceedings/eckstein_journal.pdf

El-Khalil, R. (2004), "Hydan: Information Hiding in Program Binaries" [online] http://www.crazyboy.com/hydan/

Foster and Liu (2005), "Catch me, if you can," Black Hat Briefings 2005.

Garfinkel and Malan (2005), "One Big File is Not Enough: A Critical Evaluation of the Dominant Free-Space Sanitization Technique,",The 6th Workshop on Privacy Enhancing Technologies, Robinson College, Cambridge, United Kingdom, June 28 - June 30.

Geiger (2005), "Evaluating Commercial Counter-Forensic Tools," *DFRWS 2005.* http://www.dfrws.org/2005/proceedings/geiger_couterforensics.pdf

Goldschlag, Reed and Syverson (1999), "Onion Routing," *Communications of the ACM*, 42(2), pp. 39—41.

Grugq (2002), "Defeating forensic analysis on UNIX," *Phrack Magazine*, 11)6—6, 28 July [online] http://www.totse.com/en/hack/hack_attack/167627.html

Grugq (2003), "To the Art of Defiling," Black Hat Asia 2003 presentation. [online] http://opensores.thebunker.net/pub/mirrors/blackhat/presentations/bh-asia-03/bh-asia-03-grugq/bh-asia-03-grugq.pdf

Grugq (2003a), "The Design and Implementation of ul_exec" Security Focus [online] http://securityfocus.com/archive/1/348638/2003-12-29/2004-01-04/0

Grugq (2005), "The Art of Defiling," Black Hat 2005, [online] http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-grugq.pdf

Guttman (1996), "Secure Deletion of Data from Magnetic and Solid-State Memory," in The Sixth USENIX Security Symposium Proceedings, San Jose, California. July 22-25.
Henry (2006), "Anti-Forensics" (presentation). [online] http://layerone.info/2006/presentations/Anti-Forensics-LayerOne-Paul_Henry.pdf
iDefense (2005), "Ethereal OSPF Protocol Dissector Buffer Overflow Vulnerability," [online] http://labs.idefense.com/intelligence/vulnerabilities/display.php?id=349
Infoworld (2003), "ISS reports Snort vulnerability," March 4.
King et al (2006), "SubVirt: Implementing malware with virtual machines", Proceedings of the 2006 IEEE Symposium on Security and Privacy, May. [online] http://www.eecs.umich.edu/virtual/papers/king06.pdf
Li u and Brown (2006), "Bleeding-Edge Anti-Forensics," *Infosec World Conference & Expo,* MIS Training Institute.
Liu and Stach (2006), "Defeating Forensic Analysis", CEIC 2006 — Technical Lecture 1. Notes. 4 May. http://www.metasploit.com/projects/antiforensics/CEIC2006-Defeating_Forensic_Analysis.pdf [online]
McDonald, A. and Kuhn, M. (2000), "StegFS: A Steganographic File System for Linux," in A. Pfitzmann (Ed.), IH'99, LNCS 1768, pp. 463-477. [online] http://www.cl.cam.ac.uk/~mgk25/ih99-stegfs.pdf
McLeod, S. (2005) "Smart Anti-Forensics," *Forensic Focus*, May 2005. [online] http://www.forensicfocus.com/smart-anti-forensics
Mehta, N. and Clowes, S. (2003), "Shiva: Advances in ELF Binary Encryption," CanSecWest.
Metasploit (2006), "Timestomp" [online] http://www.metasploit.com/projects/antiforensics/.http://www.metasploit.com/projects/antiforensics/.
Microsoft (2006), "Disabling Last Access Time in Windows Vista to improve NTFS performance," 7 November. [online] http://blogs.technet.com/filecab/archive/2006/11/07/disabling-last-access-time-in-windows-vista-to-improve-ntfs-performance.aspx.
NIST (2006) "Guidelines for Media Sanitization," NIST Special Publication 800-88, September. [online] http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf
Pluf & Ripe (2006) "Advanced Antiforensics : SELF", Sept 2, [online] http://www.7a69ezine.org/article/Advanced_Antiforensics_:_SELF
Provos, N. (2004). "Steganography Detection with Stegdetect" [online] http://www.outguess.org/detection.php
Rudys, A. (2000). "Invited Talk: Methods for Detecting Addressable Promiscuous Devices, Mudge, @stake, summarized by Algis Rudys", *;login:*, Nov. [online] http://www.usenix.org/events/sec00/novconf.pdf
Rutkowska (2004), "Red Pill… or how to detect VMM using (almost) one CPU instruction," November [online] http://invisiblethings.org/papers/redpill.html
Sandstorm (2006)
Sani, J. (2001), "Promiscuous node detection using ARP packets," presented at Black Hat 2001 [online] http://www.blackhat.com/presentations/bh-usa-01/DaijiSanai/bh-usa-01-Sanai.ppt.
SANS (2001), Security Alert Consensus 106, Network Computing and the SANS Institute, July 19. [online] http://archives.neohapsis.com/archives/securityexpress/2001/0030.html
SMART (1996) SFF Committee Specification for Self-Monitoring, Analysis and Reporting Technology (SMART), SFF-8035i Revision 2.0, April 1.
Thompson, I., and Monroe, M (2006) "FragFS: An Advanced Data Hiding Technique," BlackHat Federal, January. [online] http://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Thompson/BH-Fed-06-Thompson-up.pdf
TrueCrypt (2006), "Frequently Asked Questions," TrueCrypt Foundation, October 7. [online] http://www.truecrypt.org/faq.php
US DoD (1995), "Cleaning and Sanitization Matrix," DOS 5220.22-M, Chapter 8, US Department of Defense. [online] www.dss.mil/isec/nispom_0195.htm
USCERT (2006), "Computer Forensics" [online] http://www.us-cert.gov/reading_room/forensics.pdf
Vrba, Z. (2003), "cryptexec: Next-generation runtime binary encryption using on-demand function extraction," Phrak 0x0b(0x3f). #0x0d of 0x14. [online] http://www.phrack.org/archives/63/p63-0x0d_Next_Generation_Runtime_Binary_Encryption.txt
Wang (2004), Huaiqing Wang and Shuozhong Wang, "Cyber Warfare: Steganography vs. Steganalysis," *Communications of the ACM* , October, Vol. 47, No. 10.
Wonko-ga (2004), "Re: The Defiler's Toolkit", Google Answers, 13 May. [online] http://answers.google.com/answers/threadview?id=345604